T.C.
KARABÜK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

GÖRÜNTÜ İŞLEME TEKNİKLERİ KULLANILARAK RENKLİ OBJE TAKİBİ YAPMA

Lisans Tezi

İbrahim YILMAZ
2011010225032

Tez Danışmanı

Yrd.Doç.Dr. Can Bülent FİDAN

KARABÜK

HAZİRAN/2016

İÇİNDEKİLER

KABUL VE ONAY

İbrahim YILMAZ tarafından hazırlanan " GÖRÜNTÜ İŞLEME TEKNİKLERİ KULLANILARAK RENKLİ OBJE TAKİBİ YAPMA" başlıklı bu tezin Lisans Bitirme Tezi olarak uygun olduğunu onaylarım. 22/08/2016

Tez Danışmanı

Yrd.Doç.Dr. Can Bülent FİDAN

Bu çalışma, jürimiz tarafından oy birliği / ~~oy çokluğu~~ ile Mekatronik Mühendisliği Anabilim Dalında Lisans Bitirme Tezi olarak kabul edilmiştir. 22/08/2016

Tez Jürisi

Başkan: Yrd.Doç.Dr. Can Bülent FİDAN .....

Üye : Yrd.Doç.Dr. Cüneyt UZUNHAC .....

Üye : Doç.Dr. İsmail Hakkı Cavgar .....

KBÜ Mühendislik Fakültesi, Mekatronik Mühendisliği Mezuniyet Komisyonu ve Bölüm Başkanlığı bu tezi Lisans Bitirme Tezi olarak onamıştır. 22/08/2016

Yrd.Doç.Dr. İbrahim ÇAYIROĞLU

Mekatronik Müh. Bölüm Bşk.

TEŞEKKÜR

Bu tez çalışmasının planlanmasında, araştırılmasında şahsımdan ilgi ve desteğini hiçbir zaman esirgemeyen, bilgi ve tecrübeleri ile çalışmamı bilimsel temeller ışığında şekillendiren danışman hocam Yrd. Doç. Dr. Can Bülent FİDAN'a sonsuz teşekkürlerimi sunarım.

Tüm eğitim hayatım ve tez çalışmalarım boyunca sabır, anlayış ve sevgileri ile her zaman yanımda olan, desteklerini hiçbir zaman esirgemeyen ve şahsıma güvenen aileme teşekkür ederim.

İbrahim YILMAZ

KARABÜK, Haziran 2016

ÖZET

Lisans Tezi

GÖRÜNTÜ İŞLEME TEKNİKLERİ KULLANILARAK RENKLİ OBJE TAKİBİ YAPMA

İbrahim YILMAZ

Karabük Üniversitesi

Fen Bilimleri Enstitüsü

Mekatronik Mühendisliği Anabilim Dalı

Tez Danışmanı

Yrd.Doç.Dr. Can Bülent FİDAN

Bu çalışmada yapılmak istenen kamera üzerinden alınan görüntünün opencv programı tarafından sağlanan görüntü kütüphanelerinde yer alan görüntü işleme algoritmalarını kullanarak gerekli filtrelemeler uygulayarak görüntünün istenen düzeye getirilip obje takibi yapabilmektir

Bu amaçla görüntü işleme kapsamında yer alan görüntü eşleme teknikleri ile arama algoritmaları birlikte kullanılmıştır. Bu çalışmada, hızlı görüntü eşleme yöntemleri ve paralel programlama tekniklerine dayanan bütünleşik bir yöntem önerilmiş ve kullanılmıştır. Önerilen yöntem çok sayıda düşük ve yüksek çözünürlüklü referans ve şablon görüntü üzerinde test edilmiştir.Elde edilen sonuçlar önerilen yöntemin eşleşen görüntüleri elde etmede başarılı olduğunu ve toplam arama süresini azalttığını göstermiştir.

Anahtar Kelimeler: Görüntü içinde görüntü arama, Görüntü işleme, Görüntü eşleme, Görüntü eşikeme, c# ,visual studio.

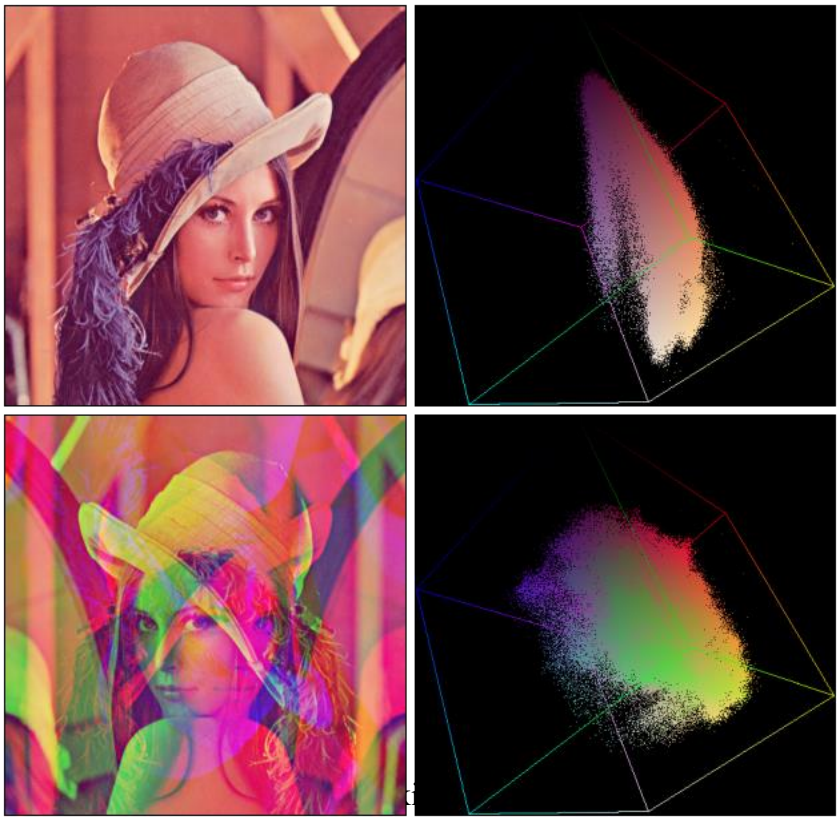
BÖLÜM 1

GİRİŞ

1.1. GÖRÜNTÜ İŞLEME

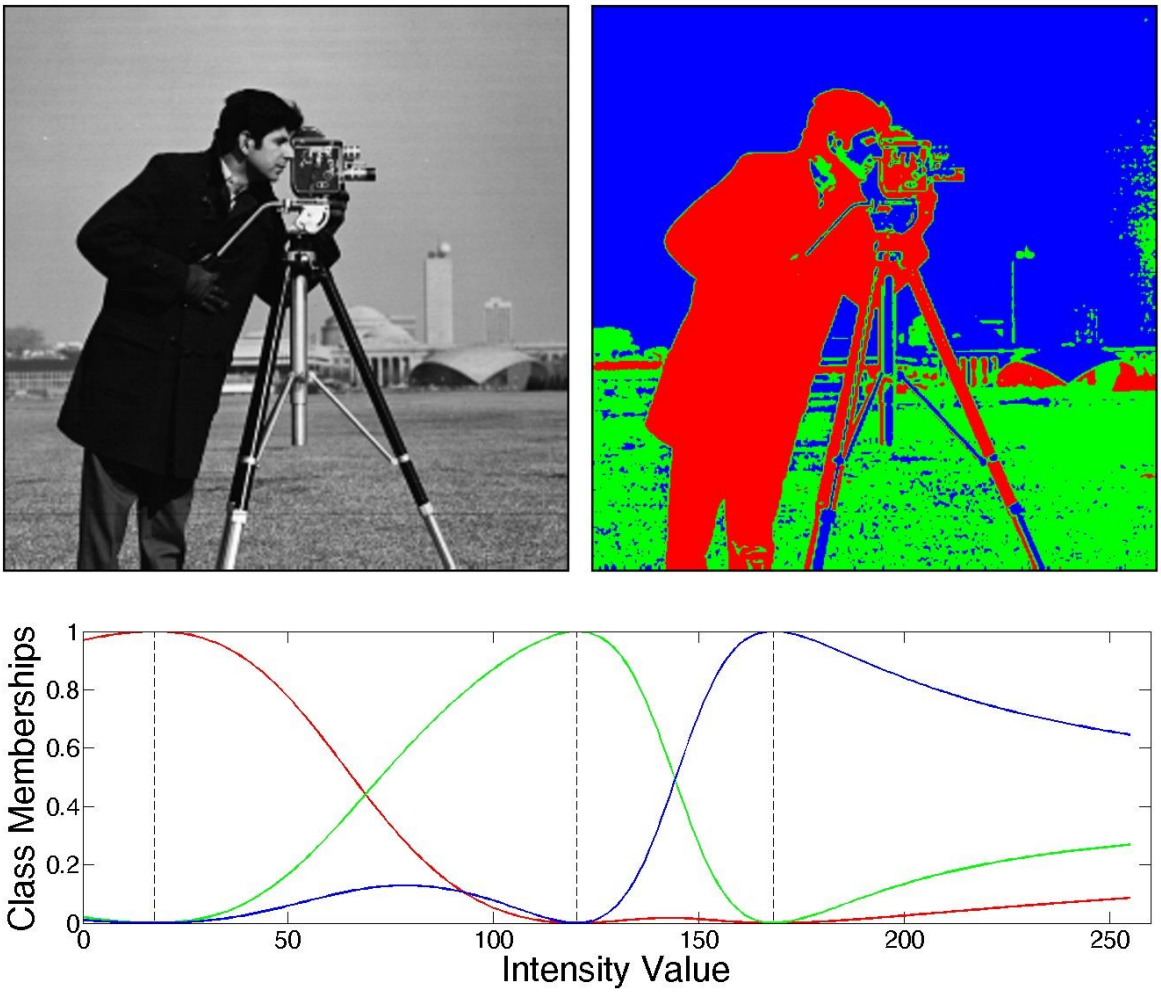
1.1.1 TANIM

Görüntü İşleme (Gİ) askeri endüstri, sualtı görüntüleme, robotik, astronomi, fizik, sanat, biyomedikal ve coğrafi bilgi sistemleri, uzaktan algılama, gözlem ve tahmin uygulamaları, hayvancılık, petrol arama, gazete ve fotoğraf endüstrisi, trafik, radar, tıp, güvenlik, suç laboratuvarları gibi pek çok alanda kullanılmaktadır [1]. Bu kapsamda Yapay Sinir Ağları (YSA), dalgacık dönüşümü, yönlendirme süzgeçleri, bulanık mantık, markov rasgele alan süzgeçleri ve görüntü iletimi gibi pek çok teknik kullanılmaktadır. Gİ'nin önemi, görüntülerin ürün pazarlama, yazıları süsleme, dijital görüntü veri tabanları, tarih, sanat, reklam, eğlence, bilim, endüstri, tıp ve uzay gibi pek çok alanda karşımıza çıkmasından kaynaklanır. Teknoloji dünyasındaki hızlı gelişmelere paralel olarak, sayısal görüntüler çok büyük bir artış göstermiş; kapsamlarının gittikçe genişlemesi ile birlikte, uygulamaya göre kullanımı değişen görüntü veri tabanlarının önemi artmıştır. Görüntüler (fotoğraf, çizim vs.), renk dağılımlarına, doku yapılarına, bölgesel şekillere, taslak uyumuna veya nesnel sınıflandırmalara göre sorgulanabilirler. İçerik tabanlı görüntü erişim sistemlerinin genel yapısı Şekil 1'de verilmiştir [2]. Bu sistemin bir parçası olarak, görüntü eşleme için tüm görüntüleri karşılaştırma ya da rasgele gözden geçirme gibi uygulamalar oldukça maliyetlidir.



Şekil 1.1

Görüntü iyileştirme ve görüntü onarmadan farklı olarak görüntü bölütleme, görüntü analizi ile ilgili bir problem olup görüntü işlemenin gösterim ve tanılama aşamalarına görüntüyü hazırlama işlemidir. Bu anlamda görüntü bölütleme, bir görüntüyü her biri içerisinde farklı özelliklerin tutulduğu anlamlı bölgelere ayırmak olarak tarif edilebilir. Bu özellikler örneğin, görüntü içerisindeki benzer parlaklıklar olabilir ve bu parlaklıklar ilgili görüntünün farklı bölgelerindeki nesneleri temsil edebilir. Görüntü içerisinde aynı parlaklıklara sahip nesne parçacıklarının belirlenmesi, sınıflandırma ve tanılama amacı için kullanılabilir. Unutulmamalıdır ki, tüm görüntülere uygulanabilecek genel (universal) bir bölütleme yöntemi yoktur ve hiçbir bölütleme yöntemi mükemmel değildir. Başka bir deyişle, görüntü iyileştirme ve onarma problemlerinde olduğu gibi görüntü bölütleme için tasarlanan yöntemler ve bu yöntemlerin başarımları, görüntüden görüntüye ve uygulamaya dayalı olarak değişiklik arz eder. Genel olarak gri-ton görüntüler için bölütleme algoritmaları, gri seviye değerlerinin iki temel özelliğinden birine dayalı olarak tasarlanırlar. Bu özellikler, görüntü içerisindeki gri seviye değerlerindeki süreksizlik (discontinuity) ve benzerlik (similarity) ile ilgilidir.



Şekil 1.2 İmaj Bölütleme

Gri seviye değerlerindeki süreksizliklere göre görüntü bölütleme problemindeki temel yaklaşım, gri seviyelerdeki ani değişiklere dayalı olarak bir görüntüyü bölmelemektir ki bu aşama bir görüntüdeki kenar ve ayrıntıların belirlenmesine (edge detection) karşı düşer.

Gri seviye değerlerindeki benzerliklere göre görüntü bölütleme ise, bölge bölütlemesi (region segmentation) olarak adlandırılır ve eşikleme (thresholding), büyütme (growing), ve yarma - kaynaştırma (split- and -merge) işlemlerine dayalı olarak gerçekleştirilir. Piksellerin gri seviye değerlerindeki benzerlik veya farklılıklara dayalı olarak bir görüntünün bölütlenmesi kavramı hem durağan (static) hem de dinamik (zamanla değişen) görüntülere uygulanabilir.

1.2. KENAR BELİRLEME

Kenar belirleme, görüntü işlemede temel öneme sahip konulardan birisidir. Bir görüntüdeki kenar, aydınlatma veya yüzey yansımaları gibi bir görüntünün fiziksel görünüşünde oluşan önemli bir değişime karşı düşer ki bu değişim kendisini parlaklık, renk ve doku olarak gösterir. Ancak burada kenar anlamında, sadece görüntü parlaklıklarındaki değişikliklerle ilgilenilecektir. Bu anlamda, bir görüntünün gri seviyelerinde ani değişikliklerin olduğu bölgelere kenar adı verilecektir. Görüntüye ilişkin kenarların belirlenmesi birçok durumda kullanışlıdır. Nesne tanıma problemi buna bir örnek olarak verilebilir. Nesne tanımada temel adım, bir görüntüyü farklı nesnelere karşı düşen farklı bölgelere

bölmelemektir. Kenar belirleme işleminin kullanışlı olduğu diğer bir örnek, sadece görüntüye ilişkin kenarların kodlandığı düşük bit oranlarında görüntü kodlama uygulamasıdır. Bir görüntüdeki fiziksel değişimin önemi, uygulamaya bağlıdır. Şöyle ki, bazı uygulamalarda kenar olarak sınıflandırılan parlaklık değişimi diğer uygulamalarda kenar olarak incelenmeyebilir. Bir nesne tanıma sisteminde nesnenin sınırları, tanıma için yeterli olabilir ve nesne içerisindeki ek ayrıntılar kenar olarak değer görmeyebilir. Sonuç olarak kenar, uygulamanın içeriği dışında tanımlanamaz. Cisimlerin fiziksel özellikleri ile kenarları arasında doğrudan bir ilişki söz konusudur. Dolayısıyla görüntünün birçok fiziksel özelliği kenar bilgisinden ortaya çıkarılabilir. Konunun önemi çerçevesinde bu bölümde kenar belirleme için çeşitli yöntemler sunulacaktır. Bunun için, görüntüye ilişkin ilgilenilen bölgelerin kendi içerisinde yeterince homojen oldukları varsayılacak ve dolayısıyla iki bölge arasındaki geçiş sadece gri seviye süreksizliklerine dayalı olarak tanımlanabilecektir. İyi bir kenar kestirimci; • kenarları iyi bir biçimde sezebilmelidir, • kenarları doğru konumlarda belirleyebilmelidir, • Bir kenar için bir kenar görüntüsü oluşturabilmeli yani yapay kenarlar üretmemelidir. Görüntü içerisinde gürültünün varlığı (gürültü yüksek frekanslı bileşenlerden oluştuğu için gürültü ile kenarları birbirinden ayırt etmek güçleşecektir), kenar belirleme ve konumlama ölçütleri arasındaki karşılıklı ilişki (tradeoff) ve kenarların çok ölçekli yapısı, kenar belirleme aşamasında karşılaşılabilecek önemli sorunlardır.

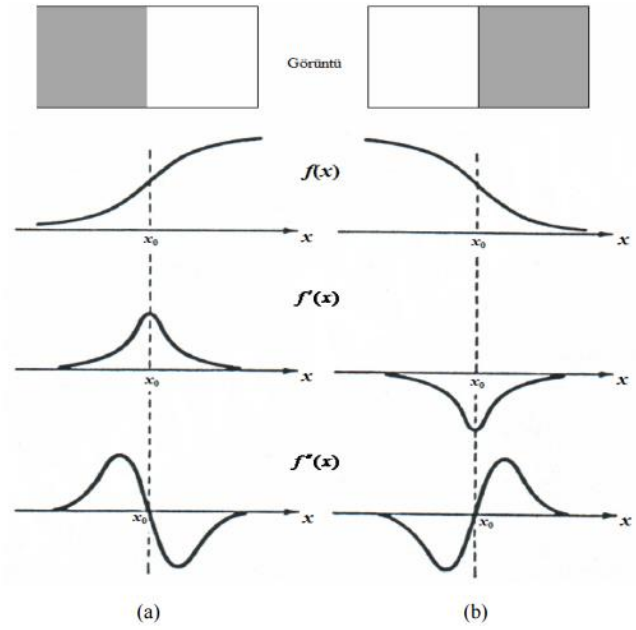
1.2.1. TÜREV ALMAYA DAYALI KENAR BELİRLEME YÖNTEMLERİ

Bir görüntü içerisindeki kenarları belirlemek için uygulanabilecek en verimli yöntemlerden birisi, ani gri seviye değişimlerini tespit etmektir. Bu amaç için birçok kenar belirleme yönteminin kullandığı temel yaklaşım, bölgesel türev hesabına dayanır. Bölgesel olarak, görüntünün 1.türevi kenar bölgelerinde en büyük değere sahip olur (local maximum) ve görüntünün 2.türevi ise kenar bölgelerinde sıfır değerini üretir. Bölgesel olarak görüntüye

ilişkin 1. ve 2. türevleri hesaplayarak elde edilen lokal maksimum ve sıfır geçiş noktaları ile, ilgili görüntü bölgesi için kenarlar belirlenmiş olur. Bu bölümde, 1.türevi kullanan gradient yöntemi ve 2.türevi kullanan laplasyen yöntemi incelenecektir.

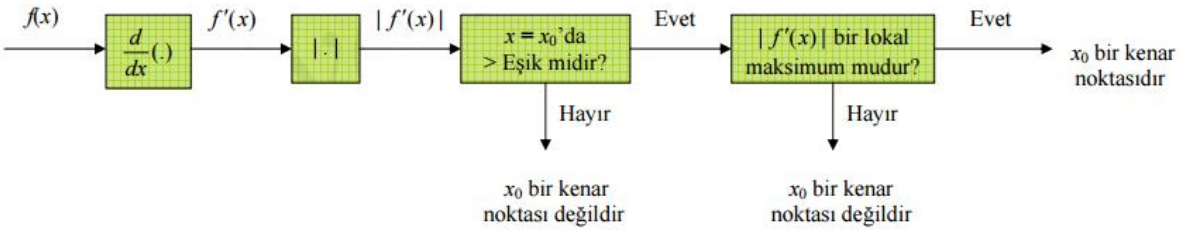
1.2.1.1. GRADİENT TABANLI YÖNTEMLER

Şekil 1.3(a)'da gösterildiği gibi tipik bir bir-boyutlu (1-B) kenarı temsil eden bir analog f(x) fonksiyonunu inceleyelim. Tipik problemlerde, şekil 4.1(a)'daki x0'ın değerini bir kenar noktası olarak incelemek mantıklıdır. x0'ı tanımlamanın bir yolu, f(x) fonksiyonunun 1.türevini (f ′(x)) veya 2.türevini (f ″(x)) hesaplamaktır. Şekil 4.1'den x0 değeri, f ′(x) 'in bölgesel ekstremumu (maksimum veya minimumu) veya f ″(x)'in sıfır geçişi (f ″(x)'in işaretini değiştirdiği nokta) aranarak tanımlanabilir. f ″(x)'in sıfır geçişi, laplasyen tabanlı yöntemler grubunda incelenecektir.



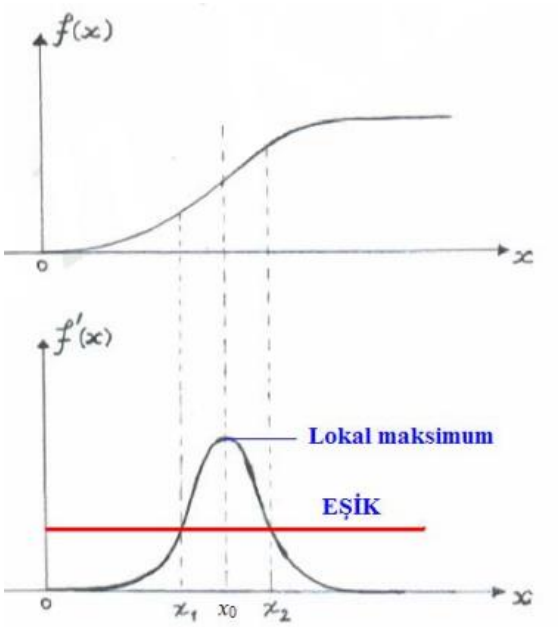
Şekil 1.4. Türev operatörleri ile kenar belirleme: (a) Koyu arka plan üzerindeki beyaz bant görüntüsü; (b) Açık arka plan üzerindeki siyah bant görüntüsü. Bu görüntülere ilişkin 1-B çizimler ve bu çizimlerin 1. ve 2.türevleri

Dikkat edilirse Şekil 1.4 (a) ve (b)'deki görüntülere karşı düşen 1-B çizimlerde kenar noktaları, ani değişimden ziyade biraz daha yumuşak bir geçiş biçiminde modellenmiştir. Bu tip modelleme, gerçekliğe yakın bir gösterim için tercih edilmiştir. Şöyle ki, örneklemenin bir sonucu olarak sayısal görüntülerdeki kenarlar genel olarak hafif bulanıklaşır. Şekil 1.4'de gösterilen muhtemel kenar noktası x0'ı tanımlamaya ek olarak, f ′(x) aynı zamanda kenarın yönünü ve büyüklüğünü kestirmede kullanılabilir. Eğer | f ′(x)| çok büyük ise, f(x) çok hızlı değişir ve bu durum parlaklıkta hızlı bir değişime karşı düşer. Eğer f ′(x) pozitif ise, f(x) artan bir fonksiyondur. Yukarıdaki gözlemlere dayalı olarak kenar belirleme için bir yaklaşım Şekil 1.5'de verilen sistemi kullanır.



Şekil 1.5. 1-B kenar belirleme sisteminin blok diyagramı

Şekil 1.5 ile verilen sistemde, ilk olarak f(x)'den | f ′(x)| hesaplanır. | f ′(x)|, belli bir eşik değerinden büyükse bu görüntü pikseli, bir kenar adayı olacaktır. İlgilenilen kenar noktasında eğer bu şart birden fazla x değerleri için sağlanırsa bu durumda bir kenar, noktadan ziyade çizgi olarak görünecektir ve kalın kenarların oluşmasına neden olacaktır. Bu sorunun üstesinden gelmek için en iyi yaklaşım, sadece | f ′(x)| değerlerinde lokal maksimuma sahip olan noktaları bulmak olacaktır ve bu noktalar kenar noktaları olarak belirlenir. Bu durum şekil 1.6'de kabaca gösterilmiştir.



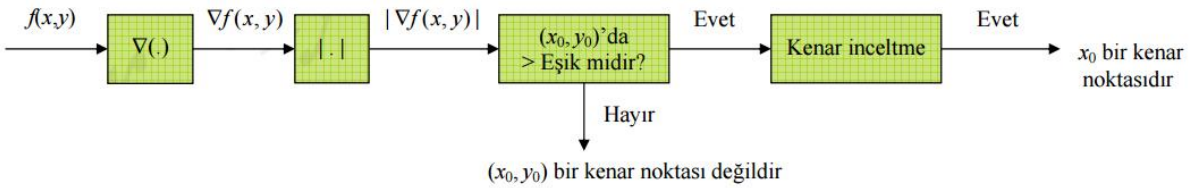
Şekil 1.6. 1-B kenar belirleme işlemleri

Kenar belirleme işleminde kullanılan eşik değerinin seçimi, uygulamaya bağlı olarak değişiklik arz eder ve kenar görüntüsünün performansı seçilen eşik değeri ile yakından ilgilidir. Eşik değeri, | f ′(x)| değerlerinin büyük olanlarını küçüklerinden ayıracak şekilde seçilmelidir. Uygun eşik değeri, | f ′(x)| değerlerinin histogramına bakılarak belirlenebilir. Ayrıca uyarlamalı olarak da eşik değerini belirlemek mümkündür. Genellikle eşik değeri, en büyük gradient %5 ile

%10'u olarak seçilebilir. 2-B bir f(x, y) fonksiyonu için f ′(x) 'in genellemesi, f(x, y)'nin gradienti olarak tanımlanır ve

(1.1)

biçiminde verilir. Burada xi r , x-yönündeki yi r ise y-yönündeki birim vektörlere karşı düşmektedir. ∇f (x, y)'ye dayalı olarak, Şekil 1.5'de verilen kenar belirleme sisteminin genellemesi Şekil 1.7'de gösterilmiştir.



Şekil 1.7. 2-B kenar belirleme sisteminin blok diyagramı

Şekil 1.7'deki sisteme göre, ilk olarak f(x, y)'nin gradienti olan ∇f (x, y) fonksiyonunun genliği hesaplanır ve bu genlik değeri daha sonra aday kenar noktalarını tanımlamak için bir eşik değeri ile karşılaştırılır. Eğer ∇f (x, y) fonksiyonunun genliğinin belirli bir eşik değerinden büyük olduğu bütün (x, y) noktaları kenar noktası olarak tanımlanırsa, oluşan kenar görüntüsü ince çizgilerden ziyade kalın şeritler halinde görünecektir. Kalın şeritlerden ince çizgiler elde etmek için, kenar görüntüsü inceltme (thinning) işlemine tabi tutulur. Basit bir kenar inceltme algoritmasında, en azından bir yönde | ∇f (x, y) | büyüklüğünün lokal maksimum olup olmadığını kontrol ederek kenar noktaları seçilir. Bir çok durumlarda, sadece yatay ve düşey yöndeki lokal maksimumu kontrol etmek yeterli olacaktır. Potansiyel bir kenar noktasında, belirlenen yönlerden herhangi biri boyunca eğer | ∇f (x, y) | lokal maksimum ise bu potansiyel kenar noktası bir kenar noktası olarak değerlendirilecektir. Diğer taraftan, bu basit kenar inceltme algoritması güçlü kenar çizgileri etrafında çok sayıda hatalı kenar çizgileri oluşturur. Bu hatalı kenar çizgilerinin çoğunu yok etmek için basit bir yöntem, aşağıdaki ek sınırlamaları sağlamalıdır: (a) Eğer yatay yönde (x0, y0) noktasında | ∇f (x, y) | lokal maksimuma sahip ise,

$$\left|\frac{\partial f(x,y)}{\partial x}\right|_{(x,y)=(x_0,y_0)} > k\left|\frac{\partial f(x,y)}{\partial y}\right|_{(x,y)=(x_0,y_0)} \qquad \textbf{(1.2)}$$

olduğunda (x0, y0) bir kenar noktası olacaktır.

(b) Eğer düşey yönde (x0, y0) noktasında | ∇f (x, y) | lokal maksimuma sahip ise

$$\left|\frac{\partial f(x,y)}{\partial y}\right|_{(x,y)=(x_0,y_0)} > k\left|\frac{\partial f(x,y)}{\partial x}\right|_{(x,y)=(x_0,y_0)} \quad \textbf{(1.3)}$$

olduğunda (x0, y0) bir kenar noktası olacaktır. (1.2) ve (1.3) eşitliklerinde k tipik olarak 2 civarında seçilir. (a) şartı sağlandığında yatay yön boyunca parlaklık değişim oranı düşey yöndekinden önemli derecede büyük olacaktır. (b) şartı sağlandığında ise düşey yön boyunca parlaklık değişim oranı yatay yöndekinden daha büyük olacaktır. Gradient tabanlı kenar belirleme sistemleri iki değişik şekilde uygulanabilir. Bunlar, yönlü ve yönsüz kenar kestirimcileri (directional and nondirectional edge detectors) olarak adlandırılır.| ∇f (x, y) |fonksiyonunu kullanan kenar belirleme sistemlerine yönsüz kenar kestirimcisi adı verilir ki bu sistem herhangi bir doğrultu için ayarlı değildir ve her yön için eşit ağırlıklı sonuçlar üretir. Eğer kenar belirleme sistemi herhangi bir yön için ayarlanmışsa bu sistemlere yönlü kenar kestirimcisi adı verilir. Örneğin, Şekil 1.4'deki kenar belirleme sisteminde | ∇f (x, y) |fonksiyonu yerine sadece | ∂f (x, y) / ∂x | fonksiyonu kullanılırsa, bu durumda sistem sadece düşey yöndeki kenarları tanır ve yatay yöndeki kenarlara ise cevap vermez. 2-B sayısal fonksiyonlarda türev işlemleri, fark denklemleri ile ifade edilebilir. Bu anlamda ∂f (x, y) / ∂x ve ∂f (x, y) / ∂y ile tanımlanan kısmi türevler fark denklemi ifadeleri ile tanımlanabilir. Örneğin, ∂f (x, y) / ∂x kısmi türevi aşağıdaki fark denklemleri ile yer değiştirilebilir:

$$\frac{\partial f(x,y)}{\partial x} \cong \frac{1}{T}[f(n_1,n_2) - f(n_1-1,n_2)] \quad (1.4a)$$

$$\frac{\partial f(x,y)}{\partial x} \cong \frac{1}{T}[f(n_1+1,n_2) - f(n_1,n_2)] \quad (1.4b)$$

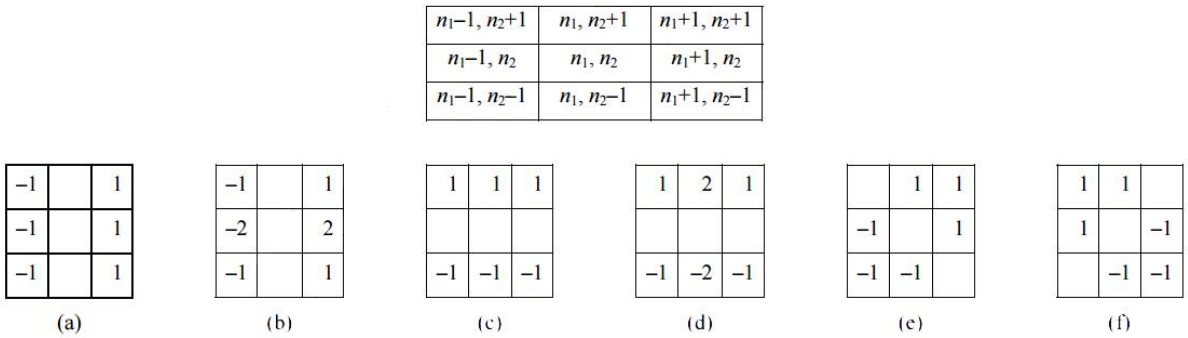
$$\frac{\partial f(x,y)}{\partial x} \cong \frac{1}{2T}[f(n_1+1,n_2) - f(n_1-1,n_2)] \quad (1.4c)$$

Kenar belirleme sisteminde, hesaplanan türev değerleri bir eşik değeri ile karşılaştırıldığından ve bu eşik değeri ayarlanabileceğinden dolayı (1.4) eşitliğindeki 1/T ve 1/2T ölçeklendirme katsayıları ihmal edilebilir. (1.4) ile tanımlanan ifadelerin daha iyi yaklaşımları için aşağıdaki gösterimler kullanılabilir:

$$\frac{\partial f(x,y)}{\partial x} \cong [f(n_1+1,n_2+1)-f(n_1-1,n_2+1)]+[f(n_1+1,n_2)-f(n_1-1,n_2)]$$
$$+[f(n_1+1,n_2-1)-f(n_1-1,n_2-1)] \quad (1.5a)$$

$$\frac{\partial f(x,y)}{\partial x} \cong [f(n_1+1,n_2+1)-f(n_1-1,n_2+1)]+2[f(n_1+1,n_2)-f(n_1-1,n_2)]$$
$$+[f(n_1+1,n_2-1)-f(n_1-1,n_2-1)] \quad (1.5b)$$

Yukarıda ifade edilen nedenden dolayı, gereksiz ölçeklendirme katsayıları (1.5) eşitliğinde ihmal edilmiştir. (1.4) ve (1.5) eşitliklerindeki fark alma işlemleri, f(n1, n2) ile impuls cevabı h(n1, n2) olan bir süzgecin konvolüsyonu olarak yorumlanabilir. Bu anlamda (1.4) ve (1.5) eşitliklerini sağlayan ve yönlü kenar kestirimcilerine karşı düşen süzgeç impuls cevapları Şekil 1.8(a) ve (b)'de gösterilmiştir. Ayrıca aynı mantıkla yatay yöndeki ve ±450 doğrultusundaki kenarları bulmak için kullanılabilecek süzgeç impuls cevapları Şekil 1.8(c)-(f)'de verilmiştir.



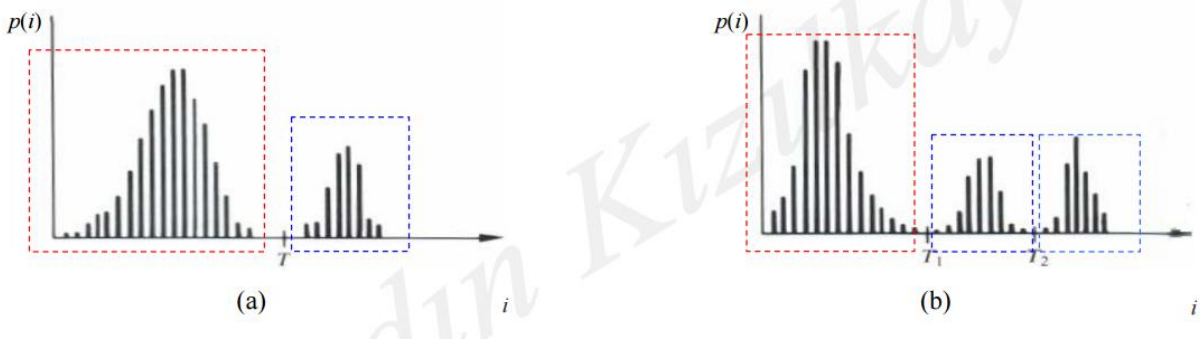
Şekil 1.8. Yönlü kenar belirleme için kullanılabilen süzgeçlerin impuls cevapları: (a) ve (b) Düşey doğrultudaki kenarlar için; (c) ve (d) Yatay doğrultudaki kenarlar için; (e) ve (f) ±450 doğrultusundaki kenarlar için.

1.2. BÖLGE BÖLÜTLEMESİ

Görüntüdeki benzerlikleri dikkate alarak görüntünün farklı bölgelere ayrılması mümkündür. Bu çerçevede görüntü bölütleme; eşikleme (thresholding), büyütme (growing), ve yarma - kaynaştırma (split- and -merge) işlemlerine dayalı olarak gerçekleştirilir.

1.2.1. EŞİKLEME

Eşikleme, görüntü bölütleme amacı için kullanılan en önemli yaklaşımlardan birisidir. Eşikleme işleminden amaç, görüntü içerisindeki nesneleri görüntü arka planından ayırmaktır. Eşikleme için, görüntüdeki gri seviye dağılımlarını gösteren görüntü histogramından faydalanılır. Örneğin, koyu bir arka plan üzerinde açık renkli nesnelerden oluşan f(i, j) görüntüsüne ilişkin gri seviye histogramı Şekil 1.13(a)'daki biçime sahip olacaktır. Bu histograma göre, nesnelere ve arka plana ait pikseller olmak üzere, görüntüyü iki ana grupta değerlendirmek mümkündür. Bu durumda nesneleri arka plandan ayırmak için en kolay yol, histogramdan göreceli olarak belirlenen bir T eşik değeri ile görüntüdeki piksel değerlerini karşılaştırmak olacaktır. Buna göre, görüntüdeki herhangi bir (i, j) pikseli için; $f(i, j) > T$ ise (i, j) pikseli nesneye ait bir nokta, $f(i, j) \leq T$ ise (i, j) pikseli arka plana ait bir nokta olacaktır. Diğer taraftan, görüntüye ilişkin histogram Şekil 1.13(b)'deki gibi ikisi nesneye biri de arka plana ait olmak üzere üç gri seviye grubundan oluşabilir. Buna göre görüntüdeki herhangi bir (i, j) pikseli için; $T1 < f(i, j) \leq T2$ aralığındaki pikseller bir nesneye, $f(i, j) > T2$ aralığındaki pikseller diğer bir nesneye ve $f(i, j) \leq T1$ aralığındaki pikseller de görüntü arka planına karşı düşecektir.



Şekil 1.13. Tek bir eşik değeri ve birden çok (çoklu) eşik değeri ile bölmelenen gri seviye histogram biçimleri

Şekil 1.13(b)'deki gibi verilen çoklu eşikleme işlemi genel olarak tek bir eşik değeri ile görüntüyü eşiklemekten daha az güvenilirdir. Bunun nedeni, birden fazla bölgeyi etkili bir biçimde bölütleyen eşik değerlerinin belirlenmesindeki zorluktur. Tipik olarak bu yapıdaki eşikleme problemleri, bölgesel olarak değişen eşik değerlerinin belirlenmesi ile çözülebilir. Bu anlamda en genel olarak eşik değeri, fonksiyon olarak

$$T = T\,[x, y, f(x,y), p(x,y)] \qquad (1.13)$$

ile verilir. Burada f(x,y), (x,y) noktasındaki gri seviye olup p(x,y) ise bu noktanın bazı bölgesel özelliklerini belirtir – örneğin bu özellik (x,y) noktasının komşuluğundaki piksel gri seviye değerlerinin ortalaması olabilir. Eşiklenmiş bir g(x,y) görüntüsü

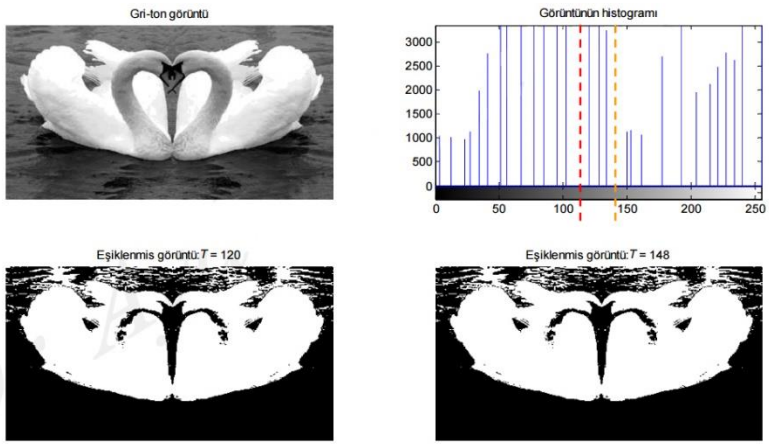
$$g(x,y) = \begin{cases} 1 & ; \quad eger \;\; f(x,y) > T \\ 0 & ; \quad eger \;\; f(x,y) \le T \end{cases} \qquad (1.14)$$

biçiminde tanımlanır. Buna göre, 1 ile etiketlenen pikseller nesneye, 0 ile etiketlenenler ise arka plana karşı düşer. Eğer ki T sadece f(x,y)'ye bağlı ise bu durumda (4.25) eşitliği ile belirlenen eşik değeri bütünsel (global) eşik olarak adlandırılır. Şekil 4.19(a), böyle bir eşik değerine örnektir. Eğer T hem f(x,y) hem de p(x,y)'ye bağlı ise bu durumda (4.25) ile belirlenen eşik değeri bölgesel (local) eşik olarak adlandırılır. Buna ek olarak, Eğer T, x ve y uzaysal koordinatlarına bağlı ise bu durumda (4.25) ile belirlenen eşik değeri dinamik eşik olarak adlandırılır.

1.2.1.1. BÜTÜNSEL (GLOBAL) EŞİKLEME

Tüm eşikleme tekniklerinin en basiti, Şekil 1.13(a)'da da gösterildiği gibi tek bir eşik değeri kullanarak görüntü histogramını ve dolayısıyla görüntüyü bölütleme işlemidir.

Bu tip bölütleme, (1.14) eşitliğini dikkate alarak, görüntüdeki piksellerin nesne veya arka plan olarak etiketlendirilmesi ile sonuçlanır. Bu bölütleme yönteminin başarısı, görüntü histogramının iyi bir biçimde bölmelenmesine bağlıdır. Şekil 1.14'de bütünsel eşikleme işlemi sonucu elde edilen görüntü sonuçları verilmiştir.[3]



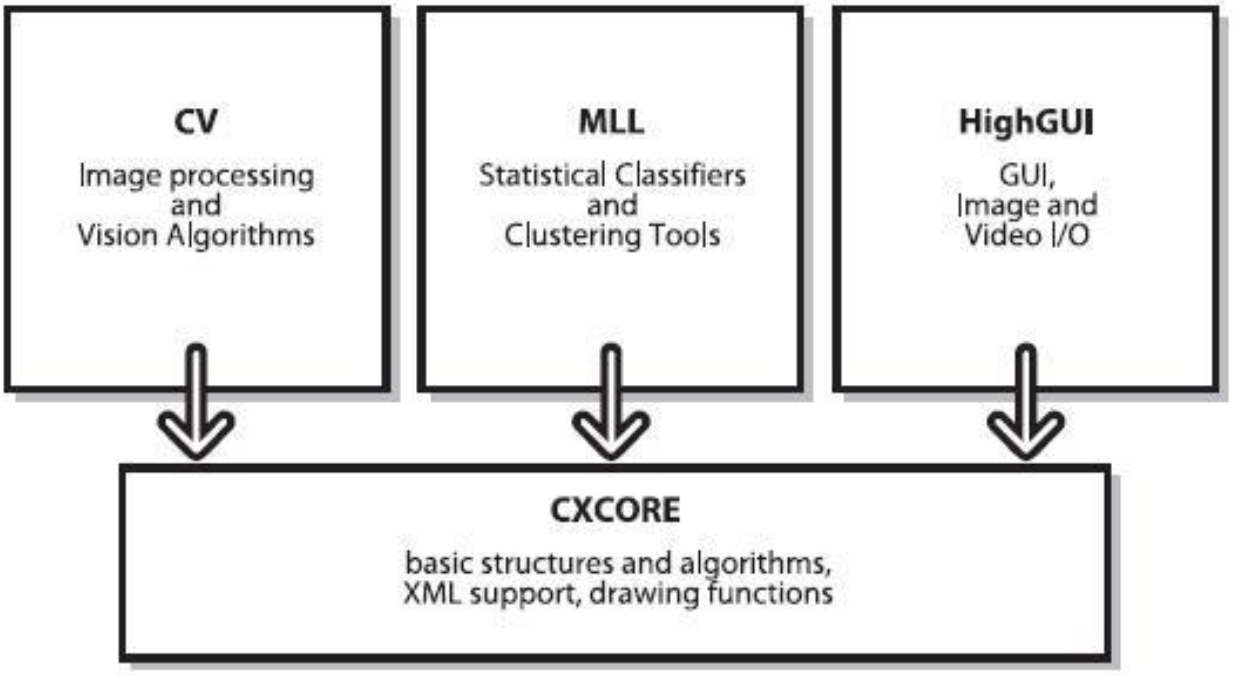
Şekil 1.14. Bütünsel eşikleme sonucu elde edilen görüntüler

BÖLÜM 2

OPENCV KÜTÜPHANESİ

2.1 OPEN CV

OpenCV, bir resim ya da video içindeki anlamlı bilgileri çıkarıp işleyebilmek için INTEL tarafından C ve C++ dilleri kullanılarak geliş- tirilmiş, açık kaynak kodlu bir "Bilgisayarla Görme" kütüphanesidir.



Şekil 2.1. OpenCV Bileşenleri[1]

OpenCV kütüphanesi, beş temel bileşenden oluşmaktadır. Bu bileşenlerin dört tanesi Şekil 2.1'de görülmektedir [4].

Computer Vision (Bilgisayarla Görü/Görme) kelimesinin baş harfleri kullanılarak isimlendirilen CV bileşeni, temel resim işleme fonksiyonları ve Bilgisayarla Görü/Görme için kullanılan yüksek seviyeli algoritmaları bünyesinde barındıran beş temel kütüphaneden biridir. Machine Learning Library kelimesinin baş harfleri kullanılarak isimlendirilen MLL bileşeni, adından da anlaşılacağı üzere Makina Öğrenmesi dalı için gerekli istatistiksel verilere ulaşmak, mevcut verileri sınıflandırmak için kullanılan fonksiyonları/araçları içeren diğer bir kütüphanedir. HighGUI bileşeni, slider, form gibi OpenCV kütüphanesi içerisinde tanımlanmış pek çok nesneyi yaratabilmemizi sağlayan bir grafik arabirimi olmakla beraber, resim ve videoları kaydetmek, yüklemek, hafı- zadan silmek için gerekli giriş/çıkış (I/O) fonksiyonlarını da içerir [4].

CXCore bileşeni, OpenCV'ye ait IplImage, cvPoint, cvSize, cvMat, cvHistogram... vs gibi veri yapılarını bünyesinde barındıran,

XML desteği de sağlayan bir kütüphanedir. Son olarak CvAux bileşeni, şablon eşleştirme (template-matching), şekil eşleştirme (shape matching), bir objenin ana hatlarını bulma (finding skeletons), yüz tanıma (face-recognition), ağız hareketleri izleme (mouth-tracking), vü- cut hareketlerini tanıma (gesture recognition) ve kamera kalibrasyonu gibi daha pek çok deneysel algoritmaları bünyesinde barındıran kütüphanedir [4].

OpenCV kütüphanesi, BSD lisansı ile lisanslanmıştır. Özgür lisanslar içinde en özgürü olarak bilinen bu lisansta kodu alan kişi, istediği gibi kullanma özgürlüğüne sahiptir [5]. Akademik ve ticari kullanımı ücretsiz olan bu kü- tüphane Windows, Linux, MacOS X gibi farklı platformlarda kullanılabilir [6].

Intel'in görüntü işleme laboratuarlarında geliştirilen ve hız açısından optimize edilen OpenCV kütüphanesi, gerçek zamanlı uygulamalar hedef alınarak geliştirilmiştir. USB 2.0 teknolojisi ile birlikte artık standart bir bilgisayarda bile gerçek zamanlı uygulamalar çalıştırılabilmektedir. Tüm bu gelişmeler oyuncak yapımından endüstriyel üretime kadar pek çok alanda bu kütüphanenin kullanılmasına yol açmıştır [7].

2.2 OPENCV İLE TEMEL UYGULAMALAR

Bu bölümde OpenCV kütüphanesini etkin bir şekilde kullanabilmek için basit örneklere yer verilmiştir. Tüm örnekler Windows 8 işletim sistemi üzerinde OpenCV_2.9.4 pre1a sürümü ve Visual Studio 2012 Yazılım Geliştirme Ortamı (IDE) kullanılarak çalıştırılmıştır.

2.3 BİR RESMİN GÖRÜNTÜLENMESİ

Aşağıdaki program dosya yolu verilen bir resmin OpenCV kütüphanesi ile nasıl açılacağını göstermektedir [4].

```
//PROGRAM 1
#include <highgui.h>
int main()
{
 IplImage* img;
 img = cvLoadImage("C:\\Lenna.jpg");
 cvNamedWindow("LENNA", 1);
 cvShowImage("LENNA", img );
 cvWaitKey(0);
 cvReleaseImage(&img);
 cvDestroyWindow("LENNA");
 return 0;
 }
```

IplImage veri yapısı, çeşitli resim dosyalarını hafızaya alabilmek için oluşturulmuş özel bir veri yapısıdır. Program 1'de istenen resim IplImage tipinde bir değişkene cvLoadImage fonksiyonu yardımıyla atanır. Bu fonksiyon, BMP, DIB, JPEG, JPG, JPE, PNG, PBM, PGM, PPM, SR, RAS, TIFF ve TIF uzantılı resim dosyalarını okuyabilir [8]. Okunabilen resim dosyaları cvShowImage Fonksiyonu ile form üzerinde gösterilir. İşlem bittikten sonra yaratılan bütün nesneler hafızadan silinir.

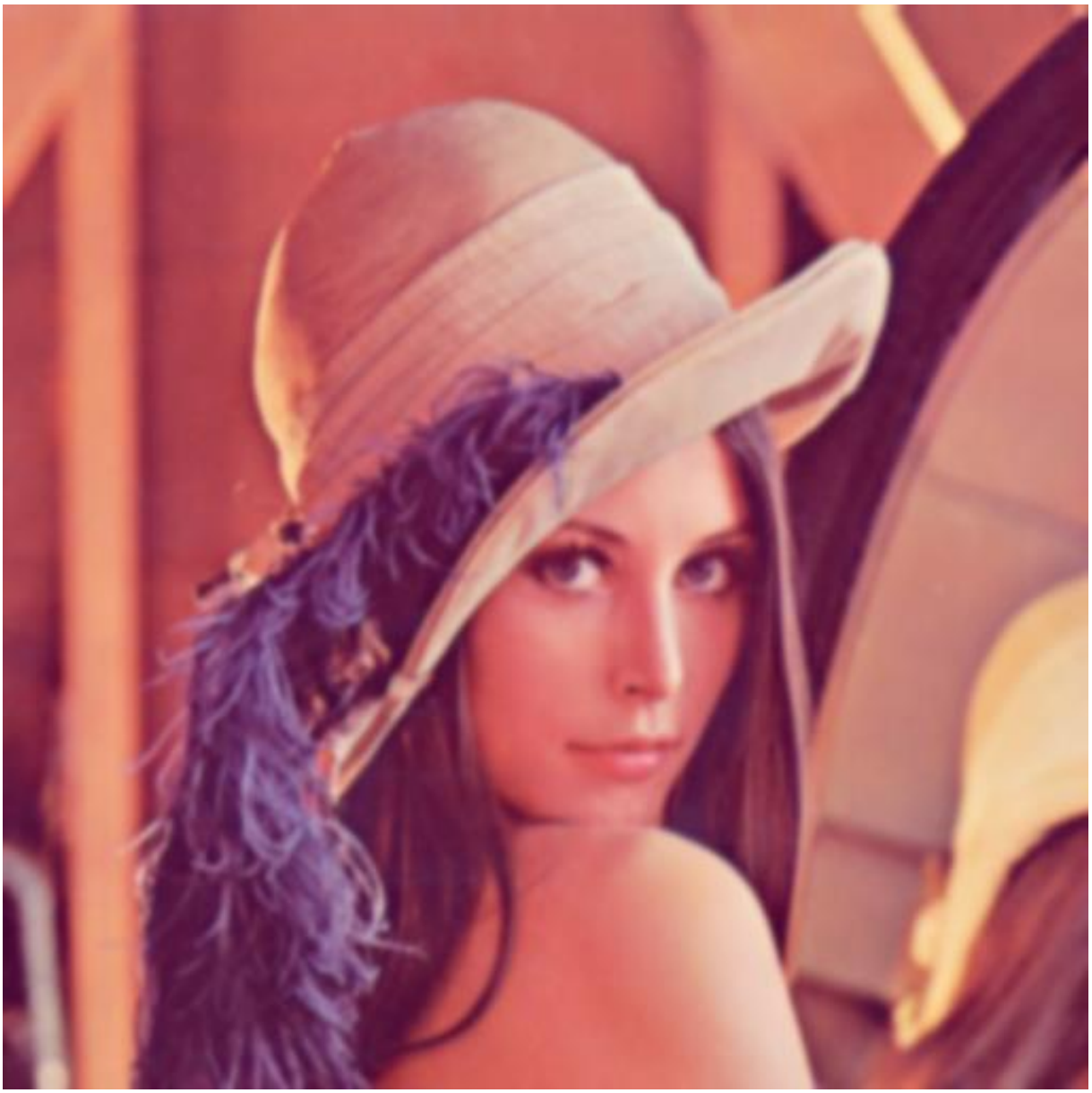
Şekil 2.2. Lenna – Standart Test İmajı

Şekil 2.2'de Program 1 çalıştırıldığında elde edilen sonuç görülmektedir. Belirtilen resim dosyası okunamadığı durumlarda aşağıdaki kontroller ve işlemler yapılmalıdır.

2.3.1 RESİMLERE FİLTRE UYGULAMA

Aşağıdaki programda, seçilen bir resme Gaussian Filtresi uygulanmıştır.

```
//PROGRAM 2
#include "cv.h"
#include "highgui.h"
int main()
{
 IplImage* img;
 img = cvLoadImage("C:\\Lenna.jpg");
 cvNamedWindow("Filtre_Oncesi",1);
 cvShowImage( "Filtre_Oncesi",img);
 cvNamedWindow("Filtre_Sonrasi",1);
 IplImage* img2;
 CvSize boyut;
 boyut=cvGetSize(img);
 img2 = cvCreateImage(boyut,8,3);
 cvSmooth(img,img2,CV_GAUSSIAN,5,5);
 cvSmooth(img2,img2,CV_GAUSSIAN,5,5);
        //Filtre Seçenekleri
        //0 CV_BLUR_NO_SCALE
        //1 CV_BLUR
        //2 CV_GAUSSIAN
        //3 CV_MEDIAN
        //4 CV_BILATERAL
 cvShowImage("Filtre_Sonrasi",img2);
 cvWaitKey(0);
 cvReleaseImage(&img);
 cvReleaseImage(&img2);
 cvDestroyWindow("Filtre_Oncesi");
 cvDestroyWindow("Filtre_Sonrasi");
 return 0;
}
```



Şekil 2.3. Gaussian Filtresi Uygulanmış Resim

Programda “cvsmooth” fonksiyonunun parametreleri değiştirilerek resme farklı filtreler de uygulanabilir.

2.3.2 KENAR BULMA UYGULAMALARI

Bütün kenar bulma işlemlerinde verilen resim önce Gri-Tonlu (Gray-Scale) resme çevirilir daha sonra istenen alt ve üst eşik değerlerine göre resim üzerindeki kenar noktaları tespit edilir. Kenar bulma yöntemlerinden bazıları Canny, Sobel ve Laplace Kenar Bulma (Edge Detector) yöntemleridir. OpenCV kütüphanesi, Canny kenar bulma yöntemi için, ‘cvCanny’, Sobel kenar bulma yöntemi için ‘cvSobel’ ve Laplace kenar bulma yöntemi için, ‘cvLaplace’ isimli

fonksiyonları içermektedir. Bu fonksiyonlara gerekli parametreler verilerek, resimdeki kenarlar tespit edilebilir. Program 3'te bu fonksiyonların kullanımı gösterilmiştir.

```
//PROGRAM 3
#include <cv.h>
#include <highgui.h>
#include <stdio.h>
int main( )
{
int sec;IplImage *rgb, *gry;
rgb=cvLoadImage( "C:\\Lenna.jpg");
cvNamedWindow("Kaynak Resim",1);
cvShowImage("Kaynak Resim",rgb);
CvSize boyut=cvGetSize(rgb);
gry = cvCreateImage(boyut,8,1);
cvCvtColor(rgb,gry,CV_RGB2GRAY);
cvNamedWindow("Gri-Tonlu Resim",1);
cvShowImage("Gri-Tonlu Resim",gry);
cvWaitKey(0);
printf("1-Canny,2-Sobel,3-Laplace");
printf("\nYontem Seciniz...\n");
scanf("%d",&sec);
if (sec==1){ //Canny
IplImage* cny;
cny = cvCreateImage(boyut,8,1);
cvNamedWindow("Canny Uygula",1);
cvCanny(gry,cny,45,120,3);
cvShowImage("Canny Uygula",cny );
cvWaitKey(0);
cvReleaseImage(&cny);
cvDestroyWindow("Canny Uygula");}
else if (sec==2){ //Sobel
IplImage* sbl;
rgb=cvCloneImage(gry);
sbl =
cvCreateImage(boyut,IPL_DEPTH_16S,1);
cvSobel (rgb, sbl, 1, 0, -1);
cvConvertScaleAbs (sbl, gry);
cvNamedWindow("Sobel Uygula",1);
cvShowImage("Sobel Uygula",gry);
cvWaitKey(0);
cvReleaseImage(&sbl);
cvDestroyWindow("Sobel Uygula");}
else if (sec==3){ //Laplace
IplImage* lplc;
lplc = cvCreateImage(boyut,IPL_
DEPTH_16S,1);
cvLaplace (gry, lplc, 3);
cvConvertScaleAbs (lplc, gry);
cvNamedWindow("Laplace Uygula",1);
cvShowImage("Laplace Uygula",gry);
cvWaitKey(0);
cvReleaseImage(&lplc);
cvDestroyWindow("Laplace Uygula");}
else printf("Yanlis Girdiniz.");
cvWaitKey(0);
cvReleaseImage(&rgb);
cvReleaseImage(&gry);
cvDestroyWindow("Kaynak Resim");
cvDestroyWindow("Gri-Tonlu Resim");
return 0;
}
```

Şekil 2.4. Gri Tonlu Resim

Şekil 2.5. cvCanny Fonksiyonu Çıktısı

Program 3, Şekil 2.2'deki standart test imajı (Lenna.jpg) ile çalıştırıldığında, Şekil 2.4'teki Gri tonlu resim ile program akışındaki seçime bağlı olarak bu üç kenar bulma yöntemlerinden biri uygulanmış resim elde edilir. Şekil 2.5'te Canny kenar bulma yöntemi ile kenarları tespit edilen resim, Şekil 2.6'da Sobel kenar bulma yöntemi ile kenarları tespit edilen resim, Şekil 2.7'de ise Laplace kenar bulma yöntemi ile kenarları tespit edilen resim görülmektedir.

Şekil 2.6. cvSobel Fonksiyonu Çıktısı

Şekil 2.7. cvLaplace Fonksiyonu Çıktısı

2.4 KAMERA İLE GERÇEK ZAMANLI GÖRÜNTÜ YAKALAMA

OpenCV kütüphanesi ile USB ya da dahili web kamerasından görüntü yakalayarak gerçekzamanlı uygulamalar geliştirilebilir. Sistemde birden fazla kamera kullanılması durumunda gerekli kamera ID'si belirlenip ilgili metoda parametre olarak girilmelidir.

Program 4'te herhangi bir kameradan görüntü yakalayabilmek için gereken temel kodlara yer verilmiştir. "cvCaptureFromCAM" metodu kullanılan kameranın ID'sini parametre olarak alır. Sistemde tek bir USB kamera kullanılıyorsa parametre olarak 0 (CV_CAP_ANY), birden fazla kamera kullanılıyorsa 100 (CV_CAP_MIL), 200 (CV_CAP_VFW) ya da 300 (CV_CAP_FIREWIRE, CV_CAP_IEEE1394, ..vs) değerlerinden biri parametre olarak kullanılır. Sistemde kullanılan kameranın çeşidine ve sayısına göre, deneme yanılma yöntemiyle gereken parametre seçilir. Program 4'te "cvCaptureFromCAM" metodu ile "CvCapture" tipinden bir değişkene görüntü gelmesi sağlandıktan sonra sonsuz bir döngü yardımıyla yakalanan görüntünün içerisindeki çerçeveler/resimler (frame) "cvQueryFrame" metodu ile tek tek sorgulanıp okutularak ekranda gösterilir. ESC'a basıldığında görüntü yakalama işlemi sona erer.

```
//PROGRAM 4

#include "highgui.h"
#include "stdio.h"
int main( )
{
cvNamedWindow( "GORUNTU",1);
CvCapture* video=cvCaptureFromCAM(0);
if (video==NULL)
{
 printf("Dosya okunamadi..\n");
 return 0;
}
IplImage* frame;
while(1)
{
 frame=cvQueryFrame(video);
 if ( !(frame)  ) break;
 cvShowImage( "GORUNTU", frame);
 char c = cvWaitKey(30);
 if ( c == 27 ) break;
}
printf("Okuma islemi bitmistir.\n");
cvReleaseCapture( &video );
cvDestroyWindow( "GORUNTU" );
cvWaitKey(0);
return 0;
}
```

BÖLÜM 3

TAKİP MEKANİZMASI VE YAZILIM

3.1 MEKANİZMA PARÇALARI

3.1.1Temel Parçalar

3.1.1.1 Arduıno

Arduino bir G/Ç kartı ve Processing/Wiring dilinin bir uygulamasını içeren geliştirme ortamındanoluşan bir fiziksel programlama platformudur.
Arduino kartlarının donanımında bir adet Atmel AVR mikrodenetleyici (ATmega328, ATmega2560, ATmega32u4 gibi) ve programlama ve diğer devrelere bağlantı için gerekli yan elemanlar bulunur. Her Arduino kartında en azından bir 5 voltluk regüle entegresi ve bir 16MHz kristal osilator (bazılarında seramik rezonatör) vardır. Arduino kartlarında programlama için harici bir programlayıcıya ihtiyaç duyulmaz, çünkü karttaki mikrodenetleyiciye önceden bir bootloader programı yazılıdır.

Şekil 3.1 Arduıno Uno

Arduino 'nun temel bileşenleri : Arduino geliştirme ortamı (IDE), Arduino bootloader (Optiboot), Arduino kütüphaneleri, AVRDude (Arduino üzerindeki mikrodenetleyici programlayan yazılım) ve derleyiciden (AVR-GCC) oluşur.

Arduino yazılımı bir geliştirme ortamı (IDE) ve kütüphanelerden oluşur. IDE, Java dilinde yazılmıştır ve Processing adlı dilin ortamına dayanmaktadır. Kütüphaneler ise C ve C++ dillerinde yazılmıştır ve AVR-GCC ve AVR Libc. ile derlenmiştir.

Optiboot bileşeni Arduino 'nun bootloader bileşenidir. Bu bileşen, Arduino kartlarının üzerindeki mikrodenetleyicinin programlanmasını sağlayan bileşendir.

Arduino 'nun bu kadar çok tercih edilmesini sağlayan en önemli bileşen ise mikrodenetleyici konusunda detaylı bilgi sahibi olmayı gerektirmeden herkesin programlama yapabilmesini sağlayan Arduino kütüphaneleridir. Arduino kütüphaneleri, geliştirme ortamı ile birlikte gelmekte ve "libraries" klasörünün altında bulunmaktadır. Kodları inceleyerek mikrodenetleyicilerin nasıl programlandığını ve kütüphanelerin yapısını görmeniz mümkündür.Son olarak AVRDude bileşeni ise derlenen kodları programlamak için kullanılır.

1 : USB jakı
2 : Power jakı (7-12 V DC)
3 : Mikrodenetleyici ATmega328
4 : Haberleşme çipi
5 : 16 MHz kristal
6 : Reset butonu
7 : Power ledi
8 : TX / NX ledleri
9 : Led
10 : Power pinleri
11 : Analog girişler
12 : TX / RX pinleri
13 : Dijital giriş / çıkış pinleri (yanında ~ işareti olan pinler PWM çıkışı olarak kullanılabilir.)
14 : Ground ve AREF pinleri
15 : ATmega328 için ICSP
16 : USB arayüzü için ICSP

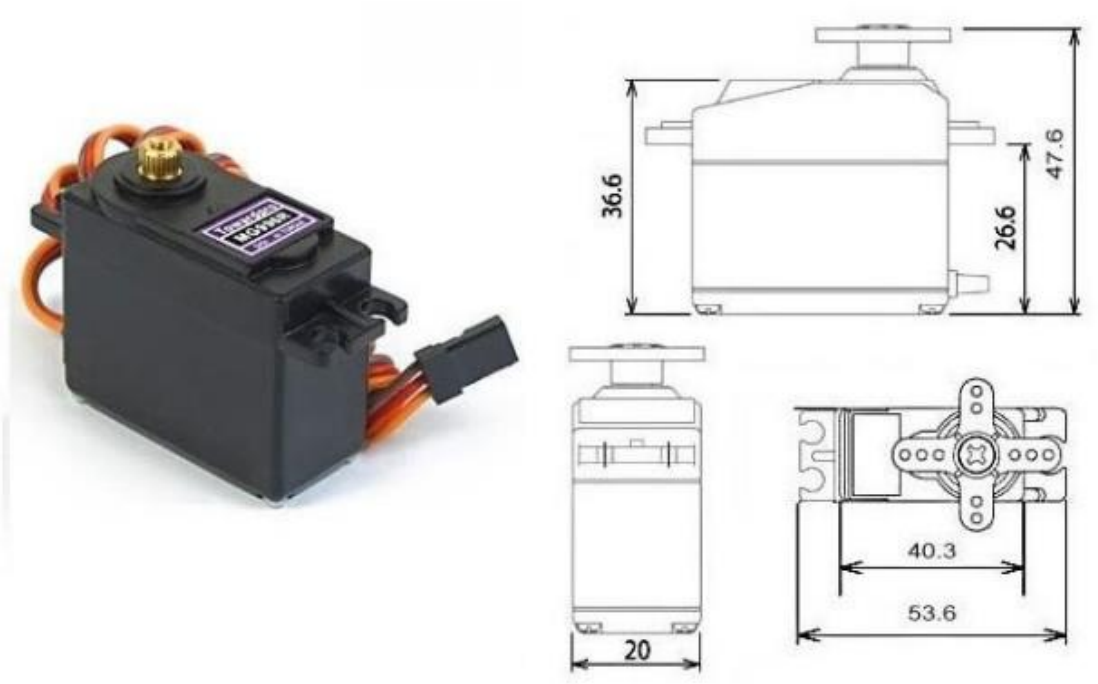
Arduino Uno Teknik Özellikleri

Mikrodenetleyici : ATmega328
Çalışma gerilimi : +5 V DC
Tavsiye edilen besleme gerilimi : 7 - 12 V DC
Besleme gerilimi limitleri : 6 - 20 V
Dijital giriş / çıkış pinleri : 14 tane (6 tanesi PWM çıkışını destekler)
Analog giriş pinleri : 6 tane

Giriş / çıkış pini başına düşen DC akım : 40 mA
3,3 V pini için akım : 50 mA
Flash hafıza : 32 KB (0.5 KB bootloader için kullanılır)
SRAM : 2 KB
EEPROM : 1 KB
Saat frekansı : 16 MHz

3.1.1.2 Servo Motor

MG996R Servo Motor; Dişli kutusu metaldir. Servo hornları ve diğer parçaları ile birlikte gönderilir. 30-160 derece arası olan bir dönme açısına sahiptir.



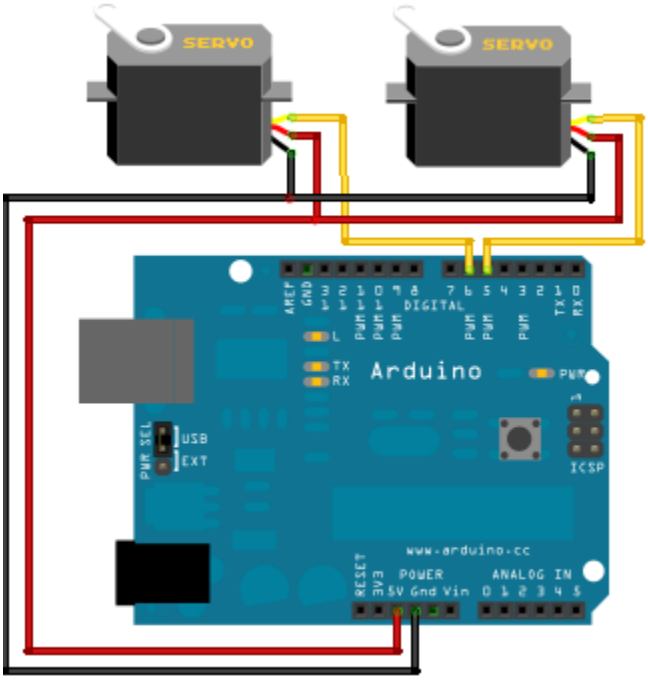
Şekil 3.2 MG996R Servo Motor

Teknik Detaylar:

Büyüklük:	40.7 x 19.7 x 42.9 mm
Ağırlık:	55 g
Hız @4.8V:	0.20 sn/60°
Zorlanma Torku @4.8V:	10 kg·cm
Zorlanma Torku @6V:	13 kg·cm

3.2. ARDUİNO GELİŞTİRME KARTI İLE SERVO SÜRMEK

Servo motor daha çok robotlarda, sanayilerde ve modelcilikte sıkça kullandığımız, girişine uyguladığımız darbe genişliği ile 0 - 180 derece açılarda dönebilen motorlardır. Bu motorlar kullanım yerlerine göre çeşitlendirilebilirler. Otomasyonda kullanılan yüksek torklu servolar olduğu gibi modelcilikte kullanılan mikro servo motorlara kadar çok fazla çeşidi bulunmaktadır. RC (Radio Control), Hobi servoları olarak da geçebilir. Arduino'nun kütüphanelerinde servo sürmek için bir servo kütüphane bulunmaktadır. Servo sürmek için PWM sinyalini Arduino içerisindeki servo kütüphanesi yardımıyla oluşturulacaktır. Şekil 7'de servo motorların Arduino'ya nasıl bağlandığı gösterilmiştir.



Şekil 3.3 Servo motorların Arduino'ya bağlantısı

Servolardan biri X düzleminde diğeri ise Y düzleminde hareket etmektedir. Görüntü işleme sonucunda algılanması gereken renkli hedefin koordinat bilgileri seri olarak Arduino'ya gönderilir. Arduino üzerindeki yazılım sayesinde servoları gereken koordinat bilgilerine göre hareket ettirir.

3.3. GÖRÜNTÜ İŞLEME YAZILIMI YARDIMIYLA NESNE TAKİBİ

OpenCV ile algılanan hedefin kamera görüntüsü düzleminde X ve Y koordinatları belirlenmiştir. Belirlenen koordinatlar posX ve posY değişkenlerine gönderilmiştir. Bu değişkenleri USB ile göndermemiz için öncelikle bir mantık kontrolüyle değerlerin değişip değişmeyeceğini kontrol etmemiz gerekir. Eğer posX ve posY değişkenleri değişmiyorsa servo motorlarımız hareket etmeyeceğinden bu bilginin sürekli gönderilmesine gerek yoktur. Yani hedefimiz hareketsiz konumdaysa hedef koordinatları gönderilmeyecektir.

Hedefimiz ne zaman hareket etmeye başlarsa hareketin algılandığı bölgenin koordinat bilgileri posX ve posY değişkenlerine atanacağından bu bilgileri USB aracılığıyla Arduino'ya ve Arduino ile servo motorlara gönderilecektir. Arduino'ya USB olarak bilgi göndermek için aşağıdaki kod kullanılır.

```
HANDLE hSerial = CreateFile(L"COM4", GENERIC_READ | GENERIC_WRITE, 0, 0,
OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);
DWORD btsIO;
```

Buradaki kodun yaptığı iş seri portu yazma durumuna açmak için bir serial işaretçisi tanımlanır ve kullanılacak port girilir. İkinci argüman ise yapılacak işi gösterir. Burada porta bilgi göndereceğimizden "w" (write) olarak ayarlanmaktadır. Bir başka husus ise servo motorlarımızın 180 derece dönme kabiliyeti olduğundan 0-180 arası değerler alması gerekmektedir. Kamera çözünürlüğü X'de 640 ve Y'de 480 olduğundan bu değerleri 180'e göre modunu alıp göndermekteyiz. Arduino seri bilgiyi aşağıdaki komut ile okuruz.

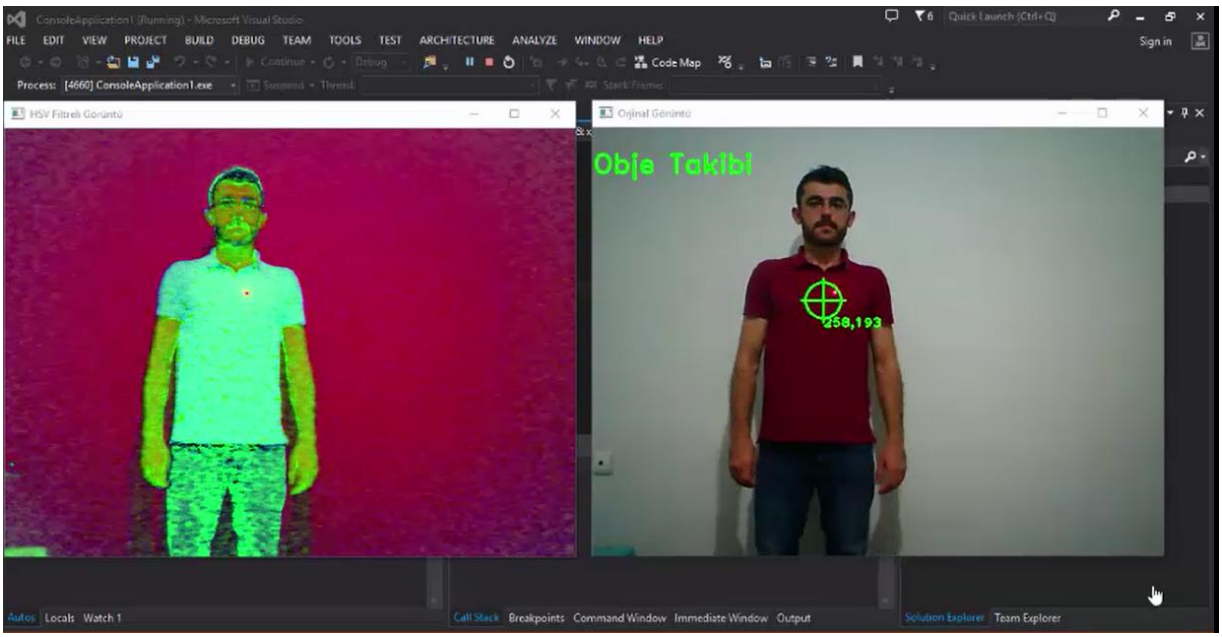
```
if (hSerial != INVALID_HANDLE_VALUE)
	{
		printf("Port opened! \n");

		DCB dcbSerialParams;
		GetCommState(hSerial, &dcbSerialParams);

		dcbSerialParams.BaudRate = CBR_9600;
		dcbSerialParams.ByteSize = 8;
		dcbSerialParams.Parity = NOPARITY;
		dcbSerialParams.StopBits = ONESTOPBIT;

		SetCommState(hSerial, &dcbSerialParams);
	}
	else
	{
		if (GetLastError() == ERROR_FILE_NOT_FOUND)
		{
			printf("Serial port doesn't exist! \n");
```

```
        }

        printf("Error while setting up serial port! \n");
}
```

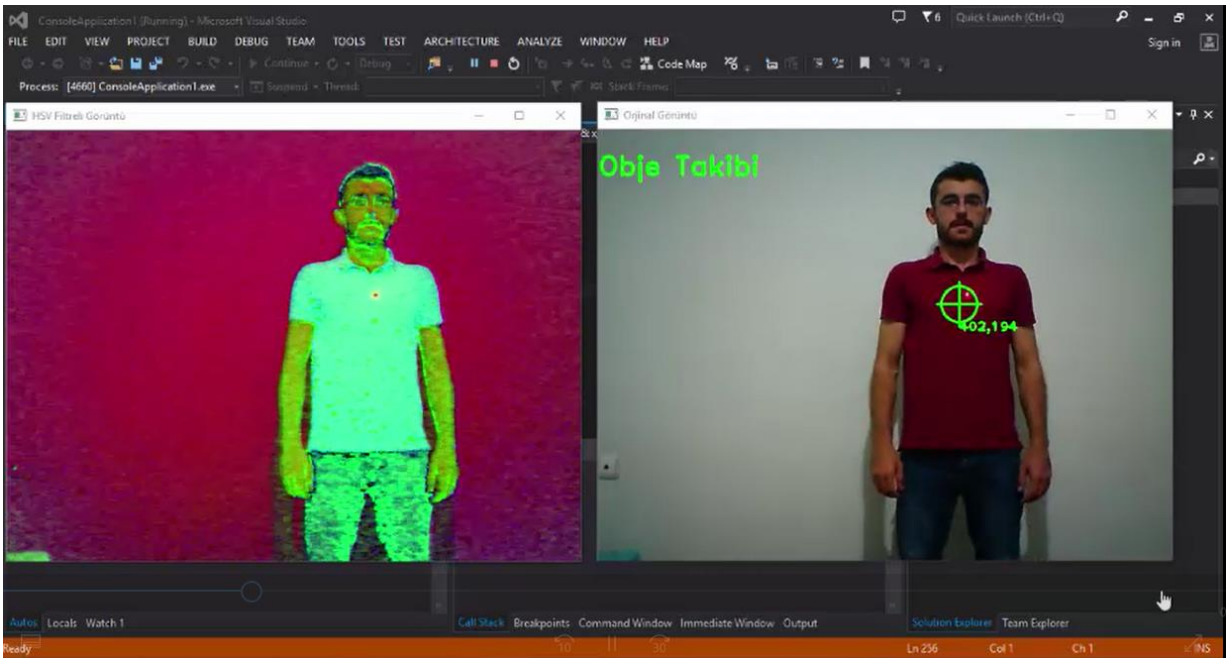
Olarak okunan seri bilgileri X ve Y koordinat bilgilerini alırız. Aldığımız bu bilgilerini Arduino'ya bağlı servo motorlara iletmek için aşağıdaki komut kullanılır.

```
v = x / 7, 11;
round(v);
printf("  Servo 1: %5d \n", v);

outputChars[0] = v;
WriteFile(hSerial, outputChars, strlen(outputChars), &btsIO, NULL);
```

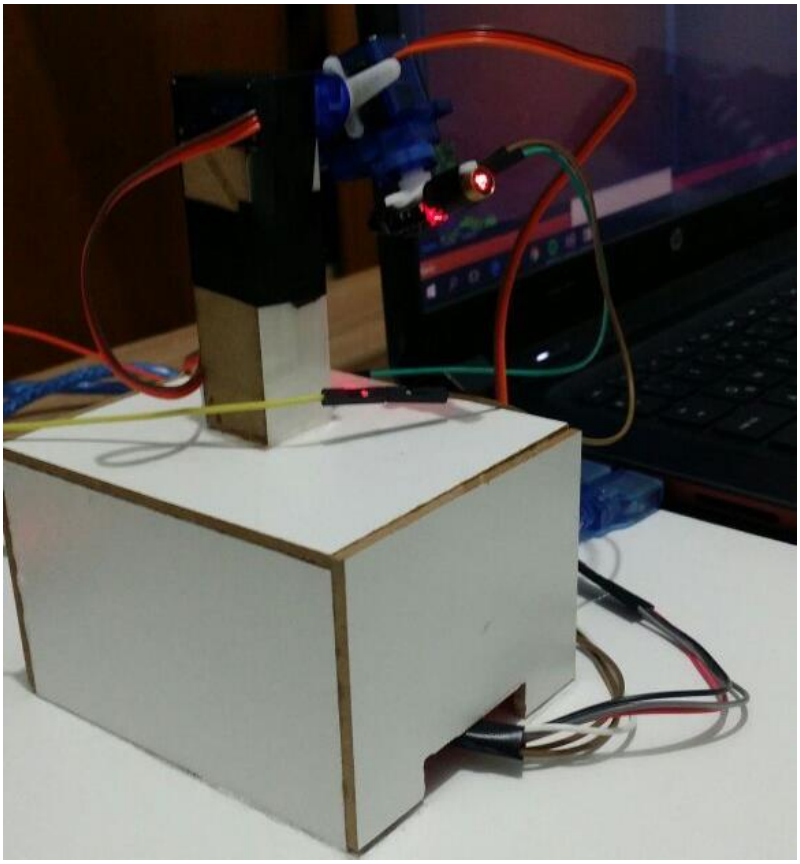
Gönderilen koordinat bilgileri servo için 0-60 derece arasında dönme hareketi yapar. Servo motorlara gelen posX bilgisi X yönünde hareket eden servo motora kaç derece dönmesi gerektiğini söyler. Aynı şekilde gelen posY bilgisi de Y yönünde hareket eden servo motora kaç derece dönmesi gerektiğini söyler. Servo motorlarımızın Arduino geliştirme kartı aracılığıyla kontrolü kısaca bu şekildedir. [9]



Şekil 3.4 Hareketli görüntüde renk takibi 1



Şekil 3.5 Hareketli görüntüde renk takibi 2



Şekil 3.6 Takip Mekanizması

BÖLÜM 4

SONUÇ VE DEĞERLENDİRME

Sonuç olarak bu çalışmada görüntü işleme ile renkli hedef takibi nasıl yapılır? Hedef takibi yaparken nelere dikkat edilir? Renk algılama nedir? Renk algılama teknikleri nelerdir? Kısaca bunlara değinilmiştir.

Yazılım kısmında renkli hedef ekranın tam orta noktasında olmasına dikkat edilmiş ve takip aşaması bu koşula göre yapılmıştır. Takip edilecek olan hedefin ağırlık merkezi görüntümüzün merkezinde olduğu için hareketteki sapmaları kontrol edilecek sisteme yani servo motorlara bilgiler gönderilmiştir.

Yazmış olduğumuz görüntü işleme yazılımına bağlı olarak çalışan donanım şekil 3.6 da gösterilmiştir. Donanım kısmı iki servonun biri X, diğeri Y yönünde hareket edecek şekilde tasarlanmıştır.

KAYNAKLAR

[1] Gonzalez RC, Woods RE. Digital Image Processing. 3rd ed. London, England, Prentice Hall, 2008.

[2] Müller H, Michoux N, Bandon D, Geissbuhler A. "A Review of Content-Based Image Retrieval Systems in Medical Applications-Clinical Benefits and fuTure Directions". International Journal of Medical Informatics, 73(1), 1-23, 2004.

[3] http://akizilkaya.pamukkale.edu.tr/B%C3%B6l%C3%BCm4_goruntu_isleme.pdf

[4] Bradski, G. and Kaehler, A., "Learning OpenCV: Computer Vision with the OpenCV Library", *O'Reilly Media*, Amerika Birleşik Devletleri, 16-17 (2008).

[5] http://www.bilisim-kulubu.com/sozluk/

[6] http://opencv.willowgarage.com/wiki/

[7] INTEL CORPORATION: Intel researchers teach computers to 'read lips' to improve accuracy of speech recognition software. M2 Presswire, Coventry, Apr 28,2003, pg1.

[8] OpenCV Reference Manuals - HighGUI Reference Manual

[9] M. Şanlı, F. Zengin, O. Urhan, M. K. Güllü, "Web Kamerasıyla Hareketli Nesne Takibi" Elektronik ve Haberleşme Mühendisliği Bölümü Mühendislik Fakültesi, İzmit/KOCAELİ, 2005

EKLER

KODLAR

```
#include <sstream>
#include <string>
#include <iostream>
#include "opencv\highgui.h"
#include "opencv\cv.h"
#include <Windows.h>

using namespace cv;
using namespace std;
HANDLE hSerial = CreateFile(L"COM5", GENERIC_READ | GENERIC_WRITE, 0, 0,
OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);
DWORD btsIO;
char outputChars[] = "0";
int v;
int z;

int H_MIN = 0;
int H_MAX = 256;
int S_MIN = 0;
int S_MAX = 256;
int V_MIN = 0;
int V_MAX = 256;

const int FRAME_WIDTH = 640;
const int FRAME_HEIGHT = 480;

const int MAX_NUM_OBJECTS = 50;

const int MIN_OBJECT_AREA = 20 * 20;
const int MAX_OBJECT_AREA = FRAME_HEIGHT*FRAME_WIDTH / 1.5;

const string windowName = "Orjinal Görüntü";
const string windowName1 = "HSV Filtreli Görüntü";
const string windowName2 = "Eşiklenen Görüntü";
const string windowName3 = "After Morphological Operations";
const string trackbarWindowName = "Görüntü Ayarı";

bool calibrationMode;

bool mouseIsDragging;
bool mouseMove;
bool rectangleSelected;
cv::Point initialClickPoint, currentMousePoint;
cv::Rect rectangleROI;
vector<int> H_ROI, S_ROI, V_ROI;
```

```
void on_trackbar(int, void*)
{



}
void createTrackbars(){



	namedWindow(trackbarWindowName, 0);

	char TrackbarName[50];
	sprintf(TrackbarName, "H_MIN", H_MIN);
	sprintf(TrackbarName, "H_MAX", H_MAX);
	sprintf(TrackbarName, "S_MIN", S_MIN);
	sprintf(TrackbarName, "S_MAX", S_MAX);
	sprintf(TrackbarName, "V_MIN", V_MIN);
	sprintf(TrackbarName, "V_MAX", V_MAX);

	createTrackbar("H_MIN", trackbarWindowName, &H_MIN, 255, on_trackbar);
	createTrackbar("H_MAX", trackbarWindowName, &H_MAX, 255, on_trackbar);
	createTrackbar("S_MIN", trackbarWindowName, &S_MIN, 255, on_trackbar);
	createTrackbar("S_MAX", trackbarWindowName, &S_MAX, 255, on_trackbar);
	createTrackbar("V_MIN", trackbarWindowName, &V_MIN, 255, on_trackbar);
	createTrackbar("V_MAX", trackbarWindowName, &V_MAX, 255, on_trackbar);


}
void clickAndDrag_Rectangle(int event, int x, int y, int flags, void* param){

	if (calibrationMode == true){

		Mat* videoFeed = (Mat*)param;

		if (event == CV_EVENT_LBUTTONDOWN && mouseIsDragging == false)
		{

			initialClickPoint = cv::Point(x, y);

			mouseIsDragging = true;
		}

		if (event == CV_EVENT_MOUSEMOVE && mouseIsDragging == true)
		{

			currentMousePoint = cv::Point(x, y);

			mouseMove = true;
```

```
		}

		if (event == CV_EVENT_LBUTTONUP && mouseIsDragging == true)
		{

			rectangleROI = Rect(initialClickPoint, currentMousePoint);


			mouseIsDragging = false;
			mouseMove = false;
			rectangleSelected = true;
		}

		if (event == CV_EVENT_RBUTTONDOWN){

			H_MIN = 0;
			S_MIN = 0;
			V_MIN = 0;
			H_MAX = 255;
			S_MAX = 255;
			V_MAX = 255;

		}
		if (event == CV_EVENT_MBUTTONDOWN){


		}
	}

}
void recordHSV_Values(cv::Mat frame, cv::Mat hsv_frame){


	if (mouseMove == false && rectangleSelected == true){


		if (H_ROI.size()>0) H_ROI.clear();
		if (S_ROI.size()>0) S_ROI.clear();
		if (V_ROI.size()>0)V_ROI.clear();

		if (rectangleROI.width<1 || rectangleROI.height<1) cout << "Lutfen istediginiz
rengi kare icine alin" << endl;
		else{
			for (int i = rectangleROI.x; i<rectangleROI.x + rectangleROI.width;
i++){

				for (int j = rectangleROI.y; j<rectangleROI.y +
rectangleROI.height; j++){
```

```
                                        H_ROI.push_back((int)hsv_frame.at<cv::Vec3b>(j,
i)[0]);
                                        S_ROI.push_back((int)hsv_frame.at<cv::Vec3b>(j,
i)[1]);
                                        V_ROI.push_back((int)hsv_frame.at<cv::Vec3b>(j,
i)[2]);
                                }
                        }
                }

                rectangleSelected = false;


                if (H_ROI.size()>0){

                        H_MIN = *std::min_element(H_ROI.begin(), H_ROI.end());
                        H_MAX = *std::max_element(H_ROI.begin(), H_ROI.end());
                        cout << "MIN 'H' VALUE: " << H_MIN << endl;
                        cout << "MAX 'H' VALUE: " << H_MAX << endl;
                }
                if (S_ROI.size()>0){
                        S_MIN = *std::min_element(S_ROI.begin(), S_ROI.end());
                        S_MAX = *std::max_element(S_ROI.begin(), S_ROI.end());
                        cout << "MIN 'S' VALUE: " << S_MIN << endl;
                        cout << "MAX 'S' VALUE: " << S_MAX << endl;
                }
                if (V_ROI.size()>0){
                        V_MIN = *std::min_element(V_ROI.begin(), V_ROI.end());
                        V_MAX = *std::max_element(V_ROI.begin(), V_ROI.end());
                        cout << "MIN 'V' VALUE: " << V_MIN << endl;
                        cout << "MAX 'V' VALUE: " << V_MAX << endl;
                }

        }

        if (mouseMove == true){

                rectangle(frame, initialClickPoint, cv::Point(currentMousePoint.x,
currentMousePoint.y), cv::Scalar(0, 255, 0), 1, 8, 0);
        }


}
string intToString(int number){


        std::stringstream ss;
        ss << number;
        return ss.str();
}
```

```
void drawObject(int x, int y, Mat &frame){



	circle(frame, Point(x, y), 20, Scalar(0, 255, 0), 2);
	if (y - 25>0)
		line(frame, Point(x, y), Point(x, y - 25), Scalar(0, 255, 0), 2);
	else line(frame, Point(x, y), Point(x, 0), Scalar(0, 255, 0), 2);
	if (y + 25<FRAME_HEIGHT)
		line(frame, Point(x, y), Point(x, y + 25), Scalar(0, 255, 0), 2);
	else line(frame, Point(x, y), Point(x, FRAME_HEIGHT), Scalar(0, 255, 0), 2);
	if (x - 25>0)
		line(frame, Point(x, y), Point(x - 25, y), Scalar(0, 255, 0), 2);
	else line(frame, Point(x, y), Point(0, y), Scalar(0, 255, 0), 2);
	if (x + 25<FRAME_WIDTH)
		line(frame, Point(x, y), Point(x + 25, y), Scalar(0, 255, 0), 2);
	else line(frame, Point(x, y), Point(FRAME_WIDTH, y), Scalar(0, 255, 0), 2);

	putText(frame, intToString(x) + "," + intToString(y), Point(x, y + 30), 1, 1, Scalar(0,
255, 0), 2);

	printf("  XPOS: %5d", x);

	v = x / 14.22;
	round(v);
	printf("  Servo 1: %5d \n", v);

	z = y /12;

	printf("  YPOS: %5d", y);
	round(z);
	printf("  Servo 2: %5d \n", z);
	outputChars[0] = v;
		WriteFile(hSerial, outputChars, strlen(outputChars), &btsIO, NULL);

		outputChars[0] = z;
		WriteFile(hSerial, outputChars, strlen(outputChars), &btsIO, NULL);

}
void morphOps(Mat &thresh){



	Mat erodeElement = getStructuringElement(MORPH_RECT, Size(3, 3));

	Mat dilateElement = getStructuringElement(MORPH_RECT, Size(8, 8));

	erode(thresh, thresh, erodeElement);
	erode(thresh, thresh, erodeElement);
	dilate(thresh, thresh, dilateElement);
```

```
	dilate(thresh, thresh, dilateElement);



}
void trackFilteredObject(int &x, int &y, Mat threshold, Mat &cameraFeed){

	Mat temp;
	threshold.copyTo(temp);

	vector< vector<Point> > contours;
	vector<Vec4i> hierarchy;

	findContours(temp, contours, hierarchy, CV_RETR_CCOMP,
CV_CHAIN_APPROX_SIMPLE);

	double refArea = 0;
	int largestIndex = 0;
	bool objectFound = false;
	if (hierarchy.size() > 0) {
		int numObjects = hierarchy.size();

		if (numObjects<MAX_NUM_OBJECTS){
			for (int index = 0; index >= 0; index = hierarchy[index][0]) {

				Moments moment = moments((cv::Mat)contours[index]);
				double area = moment.m00;


				if (area>MIN_OBJECT_AREA &&
area<MAX_OBJECT_AREA && area>refArea){
					x = moment.m10 / area;
					y = moment.m01 / area;
					objectFound = true;
					refArea = area;

					largestIndex = index;
				}
				else objectFound = false;
}

			if (objectFound == true){
				putText(cameraFeed, "Obje Takibi", Point(0, 50), 2, 1, Scalar(0,
255, 0), 2);

				drawObject(x, y, cameraFeed);
				}
}
		else putText(cameraFeed, "Aşırı Kumlanma Ayarlama yapiniz", Point(0, 50),
1, 2, Scalar(0, 0, 255), 2);
```

```
	}
}
int main(int argc, char* argv[])
{

	if (hSerial != INVALID_HANDLE_VALUE)
	{
		printf("Port Acıldı! \n");

		DCB dcbSerialParams;
		GetCommState(hSerial, &dcbSerialParams);

		dcbSerialParams.BaudRate = CBR_9600;
		dcbSerialParams.ByteSize = 8;
		dcbSerialParams.Parity = NOPARITY;
		dcbSerialParams.StopBits = ONESTOPBIT;

		SetCommState(hSerial, &dcbSerialParams);
	}
	else
	{
		if (GetLastError() == ERROR_FILE_NOT_FOUND)
		{
			printf("Seri port yok! \n");
		}

		printf("Seri Port kurma hatasi! \n");
	}

	bool trackObjects = true;
	bool useMorphOps = true;
	calibrationMode = true;

	Mat cameraFeed;

	Mat HSV;

	Mat threshold;

	int x = 0, y = 0;

	VideoCapture capture;

	capture.open(0);
	capture.set(CV_CAP_PROP_FRAME_WIDTH, FRAME_WIDTH);
	capture.set(CV_CAP_PROP_FRAME_HEIGHT, FRAME_HEIGHT);

	cv::namedWindow(windowName);

	cv::setMouseCallback(windowName, clickAndDrag_Rectangle, &cameraFeed);
```

```
	mouseIsDragging = false;
	mouseMove = false;
	rectangleSelected = false;


	while (1){

		capture.read(cameraFeed);

		cvtColor(cameraFeed, HSV, COLOR_BGR2HSV);

		recordHSV_Values(cameraFeed, HSV);

		inRange(HSV, Scalar(H_MIN, S_MIN, V_MIN), Scalar(H_MAX, S_MAX,
V_MAX), threshold);

		if (useMorphOps)
			morphOps(threshold);

		if (trackObjects)
			trackFilteredObject(x, y, threshold, cameraFeed);

			if (calibrationMode == true){

			createTrackbars();
			imshow(windowName1, HSV);
			imshow(windowName2, threshold);
		}
		else{

			destroyWindow(windowName1);
			destroyWindow(windowName2);
			destroyWindow(trackbarWindowName);
}
		imshow(windowName, cameraFeed);


if (waitKey(30) == 99) calibrationMode = !calibrationMode
}
return 0;
}
```

ÖZGEÇMİŞ

İbrahim YILMAZ 1992'de Trabzon'da doğdu; ilk ve orta öğrenimini de Trabzon'da tamamladı, 2006-2010 yılları arası Pelitli Ahmet Can Bali Anadolu Lisesi'nde eğitimini devam ettirdi, 2010 yılında Karabük Üniversitesi Mühendislik Fakültesi- Mekatronik Mühendisliği Bölümü'nde öğrenime başladı.

İletişim Bilgileri

Adres: Adnan Kahveci Mahallesi Topal Osman Sokak CH Blok No:3

E-posta: ibrahimyilmaz92@yandex.com

Tel: +90 545 317 61 01